文档 > API 文档

基本信息

公开的服务地址

```
https://api.moonshot.cn
```

Moonshot 提供基于 HTTP 的 API 服务接入。对 Chat API，我们兼容了 OpenAI 的相关 API 的输入和输出。

快速开始

Python 调用方法

对 python 用户，可以简单复用 openai 的 sdk。

```
pip install --upgrade 'openai>=1.0'
```

> 您需要确保使用的 python 版本至少为 3.7.1， openai 的 sdk 版本不低于 1.0.0
>
> 我们可以这样简单检验下自己库的版本：
>
> ```
> python -c 'import openai; print("version =",openai.__version__)'
> # 输出可能是 version = 1.10.0，表示当前 python 实际使用了 openai 的 v1.10.0 的库
> ```

一个简单的例子如下：

```
from openai import OpenAI

client = OpenAI(
    api_key="MOONSHOT_API_KEY",
    base_url="https://api.moonshot.cn/v1",
```

```
)

completion = client.chat.completions.create(
  model="moonshot-v1-8k",
  messages=[
    {"role": "system", "content": "你是 Kimi，由 Moonshot AI 提供的人工智能助手，你更擅长中
    {"role": "user", "content": "你好，我叫李雷，1+1等于多少？"}
  ],
  temperature=0.3,
)

print(completion.choices[0].message)
```

其中 MOONSHOT_API_KEY 需要替换为您在平台上创建的 API Key。

上面的代码中语言模型将用户信息列表作为输入，并将模型生成的信息作为输出返回。 下面是一组简单的实现多轮对话的例子：

```
from openai import OpenAI

client = OpenAI(
    api_key="MOONSHOT_API_KEY",
    base_url="https://api.moonshot.cn/v1",
)

history = [
    {"role": "system", "content": "你是 Kimi，由 Moonshot AI 提供的人工智能助手，你更擅长中
]

def chat(query, history):
    history += [{
        "role": "user",
        "content": query
    }]
    completion = client.chat.completions.create(
        model="moonshot-v1-8k",
        messages=history,
        temperature=0.3,
    )
    result = completion.choices[0].message.content
    history += [{
        "role": "assistant",
        "content": result
    }]
    return result

print(chat("地球的自转周期是多少？", history))
print(chat("月球呢？", history))
```

值得注意的是，随着对话的进行，模型每次需要传入的 token 都会线性增加，必要时，需要一些策略进行优化，例如只保留最近几轮对话。

CURL 调用方法

同样的，我们也可以基于 curl 直接调用，实现一致的效果。

```
curl https://api.moonshot.cn/v1/chat/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $MOONSHOT_API_KEY" \
  -d '{
     "model": "moonshot-v1-8k",
     "messages": [
        {"role": "system", "content": "你是 Kimi，由 Moonshot AI 提供的人工智能助手，你更
        {"role": "user", "content": "你好，我叫李雷，1+1等于多少？"}
     ],
     "temperature": 0.3
   }'
```

其中 `$MOONSHOT_API_KEY` 部分需要替换为您自己的 API Key。或者在调用前给它设置好环境变量。

API 说明

Chat Completion

请求地址

```
POST https://api.moonshot.cn/v1/chat/completions
```

请求内容

示例

```
{
    "model": "moonshot-v1-8k",
```

```
    "messages": [
        {
            "role": "system",
            "content": "你是 Kimi，由 Moonshot AI 提供的人工智能助手，你更擅长中文和英文的对
        },
        { "role": "user", "content": "你好，我叫李雷，1+1等于多少？" }
    ],
    "temperature": 0.3
}
```

字段说明

字段	是否必须	说明	类型	取值
messages	required	包含迄今为止对话的消息列表	List[Dict]	这是一个结构体的列表，每个元素类似如下：`{"role": "user", "content": "你好"}` role 只支持 `system`，`user`，`assistan` 其一，content 不得为空
model	required	Model ID, 可以通过 List Models 获取	string	目前是 `moonshot-v1-8k`，`moonshot-v1-32k`，`moonshot-v1-128k` 其一
max_tokens	optional	聊天完成时生成的最大 token 数。如果到生成了最大 token 数个结果仍然没有结束，finish reason 会是	int	这个值建议按需给个合理的值，如果不给的话，我们会给一个不错的整数比如 1024。**特别要注意的是**，这个 `max_tokens` 是指您期望我们**返回**的 token 长度，而不是输入 + 输出的总长度。比如对一个 `moonshot-v1-8k` 模型，它的最大输入 + 输出总长度是 8192，当输入 messages 总长度为 4096 的时候，您最多只能设置为 4096，否则我们服务会返回不合法的输入参数（invalid_request_error），这

字段	是否必须	说明	类型	取值
		"length", 否则会是 "stop"		拒绝回答。如果您希望获得“输入的精确 token 数”，可以使用下面的“计算 Token API 使用我们的计算器获得计数
temperature	optional	使用什么采样温度，介于 0 和 1 之间。较高的值（如 0.7）将使输出更加随机，而较低的值（如 0.2）将使其更加集中和确定性	float	如果设置，值域须为 `[0, 1]` 我们推荐 0.3，以达到合适的效果
top_p	optional	另一种采样方法，即模型考虑概率质量为 top_p 的标记的结果。因此，0.1 意味着只考虑概率质量最高的	float	默认 1.0

字段	是否必须	说明	类型	取值
		10% 的标记。一般情况下，我们建议改变这一点或温度，但不建议同时改变		
n	optional	为每条输入消息生成多少个结果	int	默认为 1，不得大于 5。特的，当 temperature 非常小靠近 0 的时候，我们只能返回 1 个结果，如果这个时候 n 已经设置并且 > 1，我们服务会返回不合法的输入参数(invalid_request_error)
presence_penalty	optional	存在惩罚，介于-2.0到2.0之间的数字。正值会根据新生成的词汇是否出现在文本中来进行惩罚，增加模型讨论新话题的可能性	float	默认为 0

字段	是否必须	说明	类型	取值
frequency_penalty	optional	频率惩罚，介于-2.0到2.0之间的数字。正值会根据新生成的词汇在文本中现有的频率来进行惩罚，减少模型一字不差重复同样话语的可能性	float	默认为 0
stop	optional	停止词，当全匹配这个（组）词后会停止输出，这个（组）词本身不会输出。最多不能超过 5 个字符串，每个字符串不得	String, List[String]	默认 null

字段	是否必须	说明	类型	取值
		超过 32 字节		
stream	optional	是否流式返回	bool	默认 false, 可选 true

返回内容

对非 stream 格式的，返回类似如下：

```
{
    "id": "cmpl-04ea926191a14749b7f2c7a48a68abc6",
    "object": "chat.completion",
    "created": 1698999496,
    "model": "moonshot-v1-8k",
    "choices": [
        {
            "index": 0,
            "message": {
                "role": "assistant",
                "content": " 你好，李雷！1+1等于2。如果你有其他问题，请随时提问！"
            },
            "finish_reason": "stop"
        }
    ],
    "usage": {
        "prompt_tokens": 19,
        "completion_tokens": 21,
        "total_tokens": 40
    }
}
```

对 stream 格式的，返回类似如下：

```
data: {"id":"cmpl-1305b94c570f447fbde3180560736287","object":"chat.completion.chunk"

data: {"id":"cmpl-1305b94c570f447fbde3180560736287","object":"chat.completion.chunk"

...

data: {"id":"cmpl-1305b94c570f447fbde3180560736287","object":"chat.completion.chunk"
```

```
data: {"id":"cmpl-1305b94c570f447fbde3180560736287","object":"chat.completion.chunk"

data: [DONE]
```

调用示例

Python 流式调用

对简单调用，见前面。对流式调用，可以参考如下代码片段：

```python
import os
from openai import OpenAI

client = OpenAI(
    api_key="MOONSHOT_API_KEY",
    base_url="https://api.moonshot.cn/v1",
)

response = client.chat.completions.create(
    model="moonshot-v1-8k",
    messages=[
        {
            "role": "system",
            "content": "你是 Kimi，由 Moonshot AI 提供的人工智能助手，你更擅长中文和英文的对i
        },
        {"role": "user", "content": "你好，我叫李雷，1+1等于多少？"},
    ],
    temperature=0.3,
    stream=True,
)

collected_messages = []
for idx, chunk in enumerate(response):
    # print("Chunk received, value: ", chunk)
    chunk_message = chunk.choices[0].delta
    if not chunk_message.content:
        continue
    collected_messages.append(chunk_message)  # save the message
    print(f"#{idx}: {''.join([m.content for m in collected_messages])}")
print(f"Full conversation received: {''.join([m.content for m in collected_messages]
```

List Models

请求地址

```
GET https://api.moonshot.cn/v1/models
```

调用示例

Python 调用

```python
import os
from openai import OpenAI

client = OpenAI(
    api_key="MOONSHOT_API_KEY",
    base_url="https://api.moonshot.cn/v1",
)

model_list = client.models.list()
model_data = model_list.data

for i, model in enumerate(model_data):
    print(f"model[{i}]:", model.id)
```

CURL 调用

```curl
curl https://api.moonshot.cn/v1/models -H "Authorization: Bearer $MOONSHOT_API_KEY"
```

文件内容抽取

> 该功能可以实现让模型获取文件中的信息作为上下文。本功能需要配合文件上传等功能共同使用。

调用示例

Python 调用

```python
from pathlib import Path
from openai import OpenAI

client = OpenAI(
    api_key="MOONSHOT_API_KEY",
```

```
    base_url="https://api.moonshot.cn/v1",
)

# xlnet.pdf 是一个示例文件, 我们支持 pdf, doc 以及图片等格式, 对于图片和 pdf 文件, 提供 ocr 相
# xlnet.pdf 是一个示例文件, 我们支持 pdf, doc 以及图片等格式, 对于图片和 pdf 文件, 提供 ocr 相
file_object = client.files.create(file=Path("xlnet.pdf"), purpose="file-extract")

# 获取结果
# file_content = client.files.retrieve_content(file_id=file_object.id)
# 注意, 之前 retrieve_content api 在最新版本标记了 warning, 可以用下面这行代替
# 如果是旧版本, 可以用 retrieve_content
file_content = client.files.content(file_id=file_object.id).text

# 把它放进请求中
messages=[
    {
        "role": "system",
        "content": "你是 Kimi, 由 Moonshot AI 提供的人工智能助手, 你更擅长中文和英文的对话。你
    },
    {
        "role": "system",
        "content": file_content,
    },
    {"role": "user", "content": "请简单介绍 xlnet.pdf 讲了啥"},
]

# 然后调用 chat-completion, 获取 kimi 的回答
completion = client.chat.completions.create(
  model="moonshot-v1-32k",
  messages=messages,
  temperature=0.3,
)

print(completion.choices[0].message)
```

列出文件

> 本功能用于列举出用户已上传的所有文件。

请求地址

```
GET https://api.moonshot.cn/v1/files
```

调用示例

Python 调用

```
file_list = client.files.list()

for file in file_list.data:
    print(file) # 查看每个文件的信息
```

上传文件

> 注意，单个用户最多只能上传 1000 个文件，单文件不超过 100MB，同时所有已上传的文件总和不超过 10G 容量。如果您要抽取更多文件，需要先删除一部分不再需要的文件。

请求地址

```
POST https://api.moonshot.cn/v1/files
```

文件上传成功后，我们会开始抽取文件信息。

调用示例

Python 调用

```
# file 可以是多种类型
# purpose 目前只支持 "file-extract"
file_object = client.files.create(file=Path("xlnet.pdf"), purpose="file-extract")
```

删除文件

> 本功能可以用于删除不再需要使用的文件。

请求地址

```
DELETE https://api.moonshot.cn/v1/files/{file_id}
```

调用示例

Python 调用

```
client.files.delete(file_id=file_id)
```

获取文件信息

> 本功能用于获取指定文件的文件基础信息。

请求地址

```
GET https://api.moonshot.cn/v1/files/{file_id}
```

调用示例

Python 调用

```
client.files.retrieve(file_id=file_id)
# FileObject(
# id='clg681objj8g9m7n4je0',
# bytes=761790,
# created_at=1700815879,
# filename='xlnet.pdf',
# object='file',
# purpose='file-extract',
# status='ok', status_details='') # status 如果为 error 则抽取失败
```

获取文件内容

> 本功能支持获取指定文件的文件抽取结果。通常的，它是一个合法的 JSON 格式的 string，并且对齐了我们的推荐格式。 如需抽取多个文件，您可以在某个 message 中用换行符 \n 隔开，拼接为一个大字符串，role 设置为 system 的方式加入历史记录。

请求地址

```
GET https://api.moonshot.cn/v1/files/{file_id}/content
```

调用示例

Python 调用

```
# file_content = client.files.retrieve_content(file_id=file_object.id)
# type of file_content is `str`
# 注意，之前 retrieve_content api 在最新版本标记了 warning, 可以用下面这行代替
# 如果是旧版本，可以用 retrieve_content
file_content = client.files.content(file_id=file_object.id).text
# 我们的输出结果目前是一个内部约定好格式的 json, 但是在 message 中应该以 text 格式放进去
```

计算 Token

请求地址

```
POST https://api.moonshot.cn/v1/tokenizers/estimate-token-count
```

请求内容

estimate-token-count 的输入结构体和 chat completion 基本一致。

示例

```
{
    "model": "moonshot-v1-8k",
    "messages": [
        {
            "role": "system",
            "content": "你是 Kimi，由 Moonshot AI 提供的人工智能助手，你更擅长中文和英文的对i
        },
        { "role": "user", "content": "你好，我叫李雷，1+1等于多少？" }
    ]
}
```

字段说明

字段	说明	类型	取值
messages	包含迄今为止对话的消息列表。	List[Dict]	这是一个结构体的列表，每个元素类似如下：`json{"role": "user", "content": "你好"}` role 只支持 `system`，`user`，`assistant` 其一，content 不得为空
model	Model ID，可以通过 List Models 获取	string	目前是 `moonshot-v1-8k`，`moonshot-v1-32k`，`moonshot-v1-128k` 其一

调用示例

```
curl 'https://api.moonshot.cn/v1/tokenizers/estimate-token-count' \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $MOONSHOT_API_KEY" \
  -d '{
    "model": "moonshot-v1-8k",
    "messages": [
        {
            "role": "system",
            "content": "你是 Kimi，由 Moonshot AI 提供的人工智能助手，你更擅长中文和英文的对i
        },
        {
            "role": "user",
            "content": "你好，我叫李雷，1+1等于多少？"
        }
    ]
}'
```

返回内容

```
{
    "data": {
        "total_tokens": 80
    }
}
```

当没有 error 字段，可以取 data.total_tokens 作为计算结果

Java 示例

Java 语言也可以调用上面的每一个接口，请参考我们的开源仓库查阅具体的使用方法。

错误说明

下面是主要的几个错误

Error Type	HTTP Status Code	详细描述
invalid_authentication_error	401	鉴权失败请确认
invalid_request_error	400	这个通常意味着您输入格式有误，包括使用了预期外的参数，比如过大的 temperature，或者 messages 的大小超过了限制。在 message 字段通常会有更多解释
rate_limit_reached_error	429	您超速了。我们设置了最大并发上限和分钟为单位的次数限制。如果在 429 后立即重试，可能会遇到罚时建议控制并发大小，并且在 429 后 sleep 3 秒
exceeded_current_quota_error	429	Quota 不够了，请联系管理员加量

Last updated on March 28, 2024